Wrap-Up Report, RecSys09

좋아요구독추천AI Team

Team 소개

👍 **좋아요댓글구독알림~**

> BoostCamp AI Tech 4기
> 추천 시스템 9조, 좋아요…팀!
>
> **T4056_김찬호**
> **T4096_배성수**
> **T4171_이지훈**
> **T4196_정소빈**
> **T4210_조원삼**

우리팀의 목표

"레벨 2팀, 손발 맞추며 성장하자!"

- #협업
- #성장
- #레벨1 문제점 개선
- #소통
- #문제 해결

목차

1-1. 프로젝트 개요

▼ 프로젝트 주제

💯 **Deep Knowledge Tracing**

"지식 상태"를 추적하는 딥러닝 방법론.

추적한 "지식 상태"를 활용하여 아직 풀지 않은 미래의 문제에 대해서 맞을지 틀릴지 예측하는 태스크

▼ 활용 장비 및 재료

개발환경 (AI Stage Server)

- OS: Ubuntu 18.04.5 LTS
- GPU: Tesla V100-SXM2-32GB

협업

- Github
- Slack
- Zoom

▼ 데이터 개요

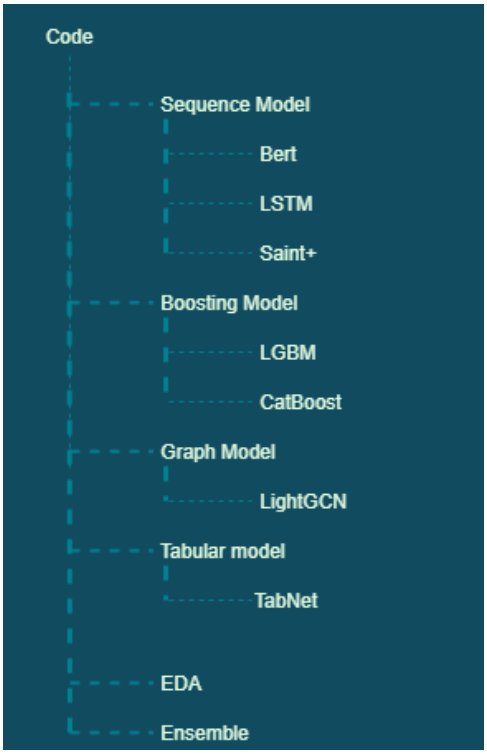
- `userID` 사용자의 고유번호. 총 7,442명의 고유 사용자가 있으며, train/test셋은 이 userID를 기준으로 90/10의 비율로 분류.
- `assessmentItemID` 문항의 고유번호. 총 9,454개의 고유 문항이 있으며, 번호 내에 카테고리, 시험지아이디등의 규칙 존재.
- `testId` 시험지의 고유번호. 시험지 내에 여러 문항이 존재합니다. 총 1,537개의 고유한 시험지가 존재.
- `answerCode` 문항을 맞췄는지 여부에 대한 정보. 0과 1은 각각 문제의 오답과 정답 여부. 타겟이 되는 마지막 문제는 -1로 지정.
- `Timestamp` 사용자가 해당문항을 풀기 시작한 시점의 데이터.
- `KnowledgeTag` 문항 당 하나씩 배정되는 태그로, 일종의 중분류 역할. 912개의 고유 태그가 존재.

이러한 형태로 약 0.9:0.1 비율로 Train/Test가 나눠져 있음.

Tools

- Python
- Pytorch
- Weights & Biases + Sweep
- Optuna

▼ 프로젝트 구조도



1-2. 프로젝트 팀 구성 및 역할

- **T4056_김찬호:**
  - 노션 프로젝트 페이지 정리
  - CatBoost 모델 적용 및 자체 EDA 진행
  - Bert 모델 적용 및 시퀀스 데이터 수정, FE 진행
  - LigthGCN 임베딩 Bert에 적용
  - Density, Histogram 기반으로 앙상블 수행
- **T4096_배성수:**
  - Bert 모델 LightGCN 임베딩 적용 및 튜닝
  - LGBM 초기모델 적용 및 EDA, FE 진행
  - TabNet 모델 적용 및 FE 진행
  - LightGCN 모델 적용 및 데이터 수정, 고도화 진행
  - Weighted, Tree, Voting 앙상블 진행
- **T4171_이지훈:**
  - Data EDA 및 Feature Engineering 진행
  - LGBM 모델링 및 Optuna 통해 모델 고도화
  - User 기반 Cross-Validation (5 & 10 fold 진행)
  - XGBM 모델링 진행
  - LSTM 모델링 및 하이퍼파라미터 튜닝 진행
- **T4196_정소빈:**
  - LGBM 모델 적용 EDA, FE 진행
  - LastQuery 모델 적용 및 FE 진행
  - LightGCN 모델 적용 및 FE 진행
- **T4210_조원삼:**
  - LightGCN 초기모델 적용 및 간단한 Tuning
  - Saint+ 모델 적용 및 코드 수정, 데이터 수정, FE 진행
  - Binary Classification을 기준으로 앙상블 진행

1-3. 프로젝트 수행 절차 및 방법

▼ EDA

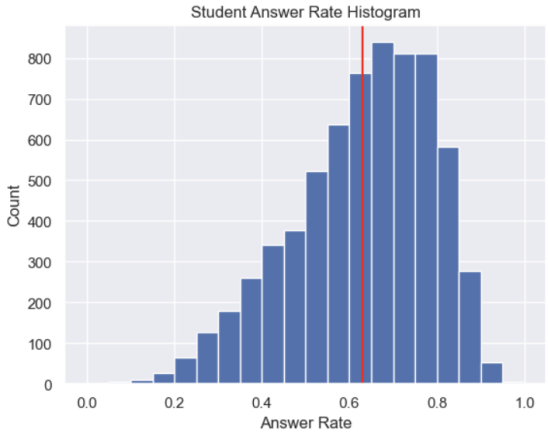
특성의 기본 정보

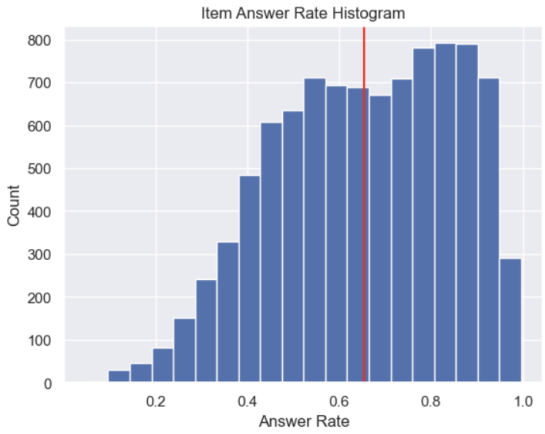
userID, assessmentItemID, testID, answerRate, KnowledgeTag와 같은 기본적인 Feature에 대한 정보를 가장 먼저 탐색하였음

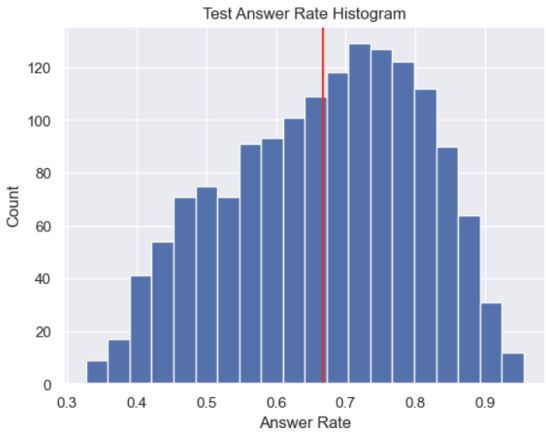
```
--- BASIC INFORMATIONS ---
userID          : 6698
assessmentItemID : 9454
testID          : 1537
mean answer rate : 65.44%
KnowledgeTag    : 912
-------------------------
```

기술 통계량 분석

유저, 문항, 시험지등을 기준으로 groupby를 진행하여 여러 피쳐들과 정답률 간의 관계에 대한 기술 통계량 분석을 진행



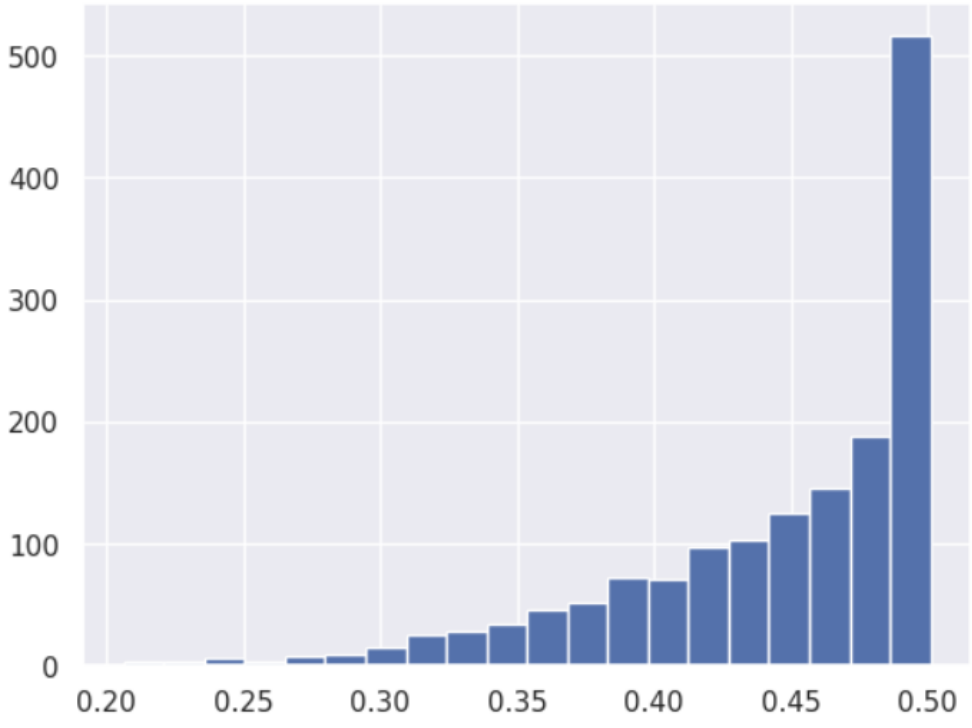




EDA

난이도 파생 변수 EDA

문항, 태그, 테스트에 대한 정답률에 대한 표준 편차가 0.5에 치우쳐있음을 확인



- 평균과 분산을 곱하여 난이도를 Quarter별로 4단계로 나눔
- 아래의 난이도 선정 기준(tag기준)
- 정답률이 낮은데, 분산도 낮다? -> 해당 tag, test가 어렵고, 보통 다 틀린다는 의미, 난이도 높음
- 정답률이 낮은데, 분산이 높다? -> 해당 tag, test가 어렵게 푸는 애도 있고, 쉽게 푸는 애도 있다, 난이도 보통
- 정답률이 높은데, 분산이 높다? -> 해당 tag, test가 어렵게 푸는 애도 있고, 쉽게 푸는 애도 있다, 난이도 보통
- 반대로, 정답률이 높은데, 분산이 낮다? -> 해당 tag, test 다 쉽게 품, 난이도 쉬움

→ 문항, 테스트, 테그 모두 적용

```
first = np.quantile(correct_k['tag_level'], 0.25)
second = np.quantile(correct_k['tag_level'], 0.5)
third = np.quantile(correct_k['tag_level'], 0.75)

correct_k.loc[correct_k['tag_level'] > third,'tag_level'] = 1
correct_k.loc[correct_k['tag_level'] <= first, 'tag_level'] = 4
correct_k.loc[ (correct_k['tag_level'] > second) & (third >= correct_k['tag_level']), 'tag_level'] = 2
correct_k.loc[(correct_k['tag_level'] > first) & (second >= correct_k['tag_level']),'tag_level'] = 3
```

```
correct_k['tag_level']
```

```
KnowledgeTag
23       3.0
24       2.0
25       2.0
26       2.0
30       3.0
        ...
11253    1.0
11265    2.0
11269    3.0
11270    4.0
11271    4.0
```

▼ 모델 탐색

Sequence Model

- **Bert**
  - 기존의 userID 기준 시퀀스는 시퀀스 데이터 개수가 적고 시퀀스별 길이의 차이가 커 제대로 학습되지 않는 문제 발생 ⇒ userID-testID를 기준으로 시퀀스를 생성하여 시퀀스 데이터 개수를 늘리고 동시에 시퀀스 길이의 차이도 줄여 학습 성능을 높임.
  - train/valid set 변경(random, 각 유저별 마지막 시퀀스, test data)
  - FE 진행: userID 추가, Timestamp 추가(numerical, split 등), LightGCN 그래프 임베딩 적용
  - Bert 모델 내부 구조 파라미터를 포함한 하이퍼 파라미터 튜닝 (with Sweep)
  - Linear warmup learning rate 스케쥴러를 사용.
- **LSTM**
  - Bert에 적용한 userID-testID 기준 시퀀스를 동일하게 적용하여 Attention 구조를 사용한 LSTM의 학습 성능을 높임.
  - Maxlength와 Layer의 수와 같은 하이퍼 파라미터 요소들을 시각화를 진행하여, 최적의 auc 성능을 내는 값으로 설정하는 튜닝을 진행
- **Saint+**

- Riiid에서 자사의 DKT문제 해결을 위해 고안한 모델로, 기존의 Saint 모델의 개량형. 실제로 Riid대회에서도 좋은 성능을 보였으며 DKT문제에 완전 적합하게 만들어졌기에 사용시, 유의미할 것으로 판단되어 사용.
- 기본적인 Saint모델의 골자는 Attention구조를 따르며, 추가적으로 DKT문제에 적합한 Time lag와 Elpased Time이라는 Features를 사용.
- 각각 Exercise에 대한 id와 Elapsed Time을 Embedding해 Encoder레이어에 넣어주고, 그에 맞는 Exercise정보를 Decoder레이어에 넣어줘 시퀀스의 마지막 문제에 대한 정답 여부를 예측. 본 대회에서는 Riiid에서 기본적으로 사용된 Feature뿐만 아니라, Item과 User의 속성 Feature를 추가해 Decoder에 전달함으로 추가적인 정보를 사용.
- Encoder에 들어갈 Features들과 Decoder에 들어갈 Features가 구분되어 있다고는 생각하지 않았기에 Item에 대한 속성은 Decoder가 아닌 Encoder에 전달하는 것을 비교해봤으나 차이는 크게 없었고 유저 속성을 넣는 것도 유의미한 차이 없음.
- 이때, 원 논문의 Backward 방식이 특이점인데, 전체 Prediction에 대해 Loss를 구하고 Backward하는것이 아닌, 라벨이 1인 경우에 대해서만 Backward를 진행했다. 이는 1을 예측하는데 더 비중을 줌과 동시에 Overfit을 방지하기 위한 목적. 다만 데이터셋의 차이로 본 대회에서는 사용하지 않는 것이 더 양호.
- Wandb Sweep을 이용한 파라미터 튜닝은 오히려 Overfit발생으로 적합하지 않음.

- **LastQuery**
  - custom transformer encoder, LSTM, DNN로 세가지 구성으로 이루어진 모델. custom transformer model은 원래의 transformer와 달리 마지막 시퀀스의 입력을 쿼리벡터로 사용하는 방식. Riiid 대회에서 긴 seq일수록 좋은 validation auc를 가졌기 때문에, 인풋 시퀀스 당 마지막 질문에 대한 정답 여부만 예측하는 학습관계(마지막 assessmentItemId[query]과 다른 assessmentItemId[key])는 충분하다고 판단.
  - custom transformer encoder에서는 모델이 assessmentItemId사이의 관계를 학습시키는 역할을 하고, LSTM에서는 sequential특징을 학습해 최근 활동에 더 큰 가중치를 둘 수 있게 함.
  - 처음 사용했던 FE는 유저별로 태그 당 정답 누적합, 태그 당 누적정답률, 전체 정답률, 최근 정답률, 푸는데 걸린시간을 사용했고, 제출한 결과와 리더보드 결과가 많이 달라 FE과정에서 과적합 문제가 생겼다고 판단.
  - 1등의 솔루션 처럼 feature를 최대한 사용하지 않는 방식으로 진행. categorical 변수로 assessmentItemId, KnowledgeTag, testId 사용했고, continuous변수로는 Elapsedtime을 이상치 처리 후 maximum value로 나눔.
  - 성능이 좋지 않아 seq_len를 늘리면서 학습시켰고, 길이를 늘리면서 성능이 떨어지는 것을 해결하기 위해 sliding window를 통해 데이터 증강 방식을 사용함.
  - Wandb Sweep을 통해 하이퍼 파라미터 튜닝을 해봤지만 적당한 하이퍼 파라미터를 결국 찾지 못했고, 이 모델은 해당 데이터셋과 맞지 않다고 판단해 사용하지 않기로 함.

Boosting Model

- **LGBM**
  - 기본 데이터 셋이 정형 데이터였고, Categorical Feature가 적고, Numerical Feature를 여러개 생성할 수 있으므로 LGBM을 사용하는 것이 유의미할 것이라고 판단.
  - Feature Engineering(FE): EDA를 통해서 User와Item(assessmentItemId, KnowledgeTag, testId)를 기준으로 정답률에 대한 Feature와 Time에 대한 Feature를 활용하여 다양한 파생 변수생성.
  - Hyper-parameter 관련 학습 후에 데이터 셋에 적합한 범위를 설정하여, 그 기반으로 Optuna를 각 fold별로 적용해서 CV(Cross-Vali
  - CV : UserID기준 Split을 진행한 것을 바탕으로 5-fold, 10-fold 진행
- **CatBoost**

- 간단한 EDA 진행: answerCode와 Timestamp를 활용한 파생 변수 생성.
- CatBoost의 Feature Importance 시각화 기능을 이용하여 이후 FE에 활용

Graph Model

- **LightGCN**
  - Learning rate를 줄이고 Epoch를 크게 설정해 학습 후 inference 결과값이 0.5에서 크게 벗어나지 않는 문제 해결.
  - 기존 baseline에서 valid set을 학습에서 사용해 valid AUC가 지속적으로 오르는 문제 발생 ⇒ 각 유저별 마지막 데이터로 valid set을 구성해 학습.
  - Sweep을 이용한 하이퍼 파라미터 튜닝.
  - LightGCN의 결과 혹은 input을 Sequence모델의 임베딩으로 받아오는 모델을 시도했으나, 시간 및 리소스 부족으로 제외.
  - 베이스 라인의 LightGCN은 모델을 임포트 해 userID의 길이와 assessmentItemId의 길이를 합친 개수를 노드 수로 사용해 임베딩 하는 방식이였고, assessmentItemId에 KnowledgeTag와 TestId 정보를 합쳐서 임베딩하기 위해 수정한 LightGCN모델을 사용.
  - userId, assessmentItemId, KnowledgeTag, TestId에 대한 임베딩 매트릭스를 따로 만들어 임베딩 시킨 벡터들을 concat 해 LGCN의 인풋으로 넣어줬지만 결과가 기존 베이스라인보다 성능이 좋지 않아 다른 정보를 합치는 과정에서 assesmentItemId에 대한 정보들이 사라져 생긴 문제라고 생각함.
  - 위 문제를 해결하기 위해 임베딩하는 과정에서 assessmentItemId의 임베딩 매트릭스에 가중치를 더 부여하는 방식을 사용.

Tabular Model

- **TabNet**
  - Test data를 valid set으로 사용해 학습해 valid와 test score간의 괴리를 최소화.
  - LGBM 베이스 라인에 사용된 Feature에 추가적으로 KnowledgeTag별 유저의 정답률과 합, 날짜, assessmentItemID 별 정답률과 합을 추가.
  - KnowledgeTag에 대한 Elo rating을 학습해 Feature로 사용.
  - 이 외에 다양한 Feature들을 추가해 봤으나 train data에 overfitting되어 적합하지 않음.

▼ 모델 고도화

하이퍼 파라미터 튜닝

- **WandB - Sweep**
  - 각 모델의 마지막 valid auc 값을 기준으로 하이퍼 파라미터를 탐색하는 문제를 해결하기 위해 best auc를 wandb_log에 추가하여 sweep 진행
  - 학습 성능에 영향을 끼치는 모델 내부 구조의 복잡도, 정규화 관련 파라미터 위주로 하이퍼 파라미터 튜닝 진행
- **Optuna & CV**
  - UserID 기준으로 5-fold, 10-fold를 진행하여 LGBM에 모델의 일반화 성능을 높이고자 진행하였음.
  - 시퀀셜 데이터이기 때문에 Data Leakage는 존재하지만, 도메인 특성상 Trend에 대한 Leakage는 영향이 없을 것이라 판단하였기에 진행
  - 각 fold별 Optuna를 적용하여 각각의 폴드에 최적의 하이퍼파라미터를 적용하였음

Ensemble

- Weighted Ensemble: LB의 AUC가 높은 모델에 가중치를 높여 적용
- density, histogram 분포 시각화 참고
- Voting Ensemble: Soft/Hard voting ensemble을 참고하여 과반수의 Label을 가진 모델들의 예측 값을 평균으로 적용
- Tree Ensemble: 각 모델의 예측 값이 0이나 1에 가까우면 값을 그대로 가져가고, 나머지는 다음 모델에 넘김. 모델 순서는 AUROC가 높은 순서로 적용. 다음 모델로 넘어갈 때 마다 패널티를 적용하여 이전 모델에서 확정된 값보다 0.5에 가까워지도록 적용.

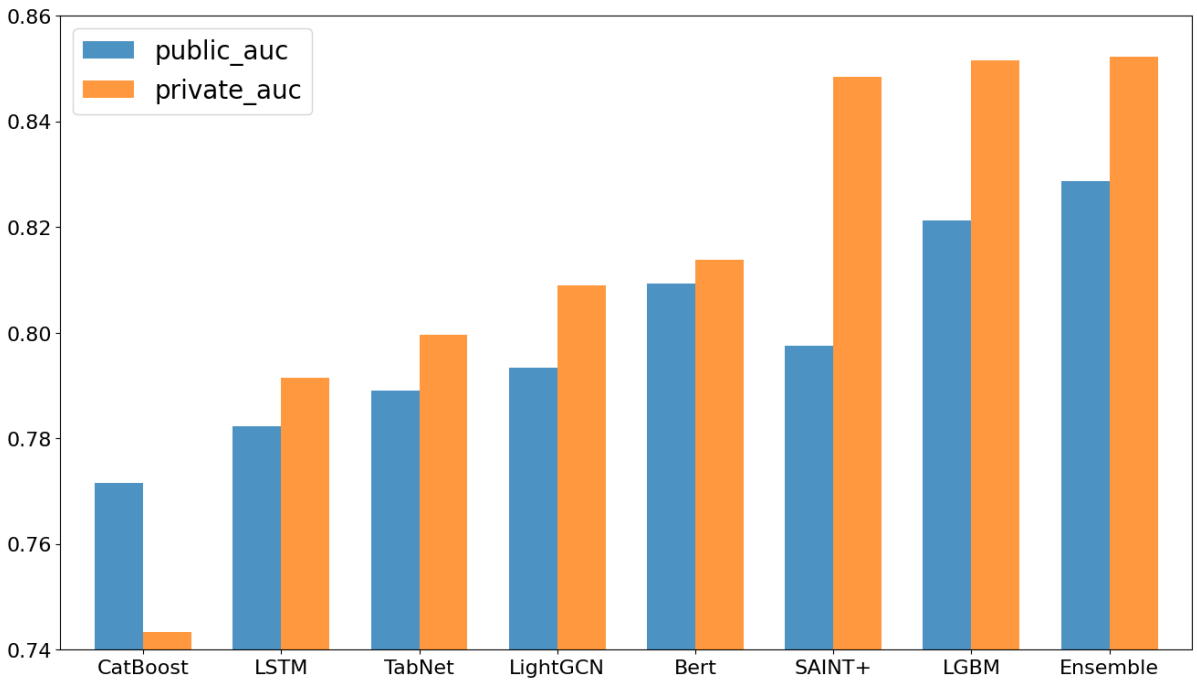
1-4. 프로젝트 로드맵

dkt 프로젝트 로드맵

Aa 이름	날짜	태그
모델 고도화 및 앙상블	@2022년 12월 5일 → 2022년 12월 8일	
랩업 리포트 작성	@2022년 12월 9일 → 2022년 12월 12일	
dkt 종료	@2022년 12월 8일	
개별 모델 탐색 및 EDA	@2022년 11월 21일 → 2022년 12월 2일	
강의 수강 및 베이스라인 이해	@2022년 11월 14일 → 2022년 11월 18일	
dkt 대회 공개	@2022년 11월 14일	
dkt 리더보드 활성화	@2022년 11월 16일	

1-5. 프로젝트 수행 결과

🥉 **최종일 리더보드 대격변! `0.8287(0.8522) / 0.7392(0.7876)` ⇒ ACC 최종 1위!**



Type	Model	Public ROCAUC	Privite ROCAUC	Public ACC	Public ACC	Ensemble
Sequence	SAINT+	0.7975	0.8484	0.7231	0.7715	X
	Bert	0.8093	0.8138	0.7446	0.7392	O
	LSTM	0.7823	0.7914	0.7124	0.7285	X
Boosting	LGBM	0.8213	0.8515	0.7473	0.7823	O
	CatBoost	0.7715	0.7433	0.7043	0.6774	X
Tabluar	TabNet	0.7891	0.7997	0.7097	0.7097	X
Graph	LightGCN	0.7934	0.8089	0.7043	0.7446	O

- **Bert**: 0.8093(0.8138) / 0.7446(0.7392), userID-testID 기준 시퀀스 사용, test data를 valid data로 사용, 간단한 하이퍼 파라미터 튜닝
- **LSTM**: 0.7823(0.7914) / 0.7124(0.7285), Attention 구조 사용, userID-testID 기준 시퀀스 사용
- **Saint+**: 0.7975(0.84840)/ 0.7231(0.7715), Feature 추가 및 코드 수정
- **LGBM**: 0.8287(0.8522) / 0.7392(0.7876), FE진행 및 userID 기반 CV 진행
- **CatBoost**: 0.7715(0.7433) / 0.7043(0.6774), 간단한 EDA 진행 후 중요도가 낮은 Feature 정리
- **LightGCN**: 0.7934(0.8089) / 0.7043(0.7446), 각 userID 별 마지막 data를 valid data로 사용, 낮은 LR과 큰 Epoch 사용
- **TabNet**: 0.7891(0.7997) / 0.7097(0.7097), Feature 추가, 간단한 하이퍼 파라미터 튜닝
- **Ensemble**: 0.8287(0.8522) / 0.7392(0.7876), LGBM, Bert, LightGCN을 Weighted 앙상블 (2:1:1)

1-6. 자체 평가 의견

▼ 잘한 점

- 다양한 모델을 구현해봤다.
  - 베이스라인 모델 수정부터 새로운 모델 구현까지
- 강의에서 나온 실험들에 더해 여러 실험들을 수행했다.
- 너무 점수에 매몰되지 않았다.
- WandB를 적극적으로 활용.

▼ 시도했으나 잘되지 않았던 점

- **TabNet**/LastQuery/Bert의 구현 및 점수가 좋지 않았다.
- Train, Test를 합친 결과가 좋지 않았는데, 그 이유를 파악하지 못했다.
- 깃헙 사용을 적극적으로 하지 못했다.

▼ 아쉬운 점

- 깃헙 사용이 제대로 되지 않았다.
- Public에만 너무 집중을 해서 제대로 모델의 성능을 최대로 끌어올리지 못했다.
- Validation에 대해, 결과적으로 좋았지만 적극적으로 CV등을 진행하지 못했다.

▼ 프로젝트를 통해 배운점

- DKT라는 Task자체를 처음으로 접해 시계열적인 데이터를 다룰 수 있는 법을 배웠다.
- Attention 구조를 많이 활용하면서 이해가 높아진 것 같다.
- 여러 모델, 기법에 대한 실험을 진행했을 때, 낙심하지 않고 다른 실험을 진행하는 것이 좋다고 생각한다.